NASA/CR-1998-208732
ICASE Report No. 98-44

On Convergence Acceleration Techniques for Unstructured Meshes

Dimitri J. Mavriplis
ICASE, Hampton, Virginia

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA

Operated by Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-97046

October 1998

ON CONVERGENCE ACCELERATION TECHNIQUES FOR UNSTRUCTURED MESHES

DIMITRI J. MAVRIPLIS *

Abstract. A discussion of convergence acceleration techniques as they relate to computational fluid dynamics problems on unstructured meshes is given. Rather than providing a detailed description of particular methods, the various different building blocks of current solution techniques are discussed and examples of solution strategies using one or several of these ideas are given. Issues relating to unstructured grid CFD problems are given additional consideration, including suitability of algorithms to current hardware trends, memory and cpu tradeoffs, treatment of non-linearities, and the development of efficient strategies for handling anisotropy-induced stiffness. The outlook for future potential improvements is also discussed.

Key words. parallel, multigrid, unstructured, Navier-Stokes

Subject classification. Applied and Numerical Mathematics

1. Introduction.

1.1. The Need for Convergence Acceleration Techniques. The field of computational fluid dynamics is generally characterized by problems of widely varying scales. For example, the use of millions of grid points, which is common today in three dimensional simulations, results in global length scales spanning the computational domain which are several orders of magnitude larger than the smallest scales resolved by neighboring grid points. Anisotropic resolution commonly employed for viscous flow calculations at high-Reynolds numbers results in streamwise length scales that are often three to four orders of magnitude larger than normal length scales. Even the time or length scales inherent in the continuous governing equations can be very disparate, such as the scales associated with acoustic and particle speeds in a nearly incompressible flow, or highly disparate reaction rates in chemically reacting flows. These phenomena all but guarantee that any simple-minded explicit scheme will be extremely inefficient for solving such problems.

While direct solution techniques can be used to solve stiff problems, several difficulties arise in the case of CFD problems. Since the governing equations are non-linear, a direct method cannot produce an answer in a single iteration (sparse matrix inversion), and must therefore be used iteratively. Although a fully converged solution can be obtained in a small number (usually O(10)) of iterations, each iteration is extremely (most often prohibitively) expensive in terms of memory and cpu requirements. This is largely due to the fact that such methods make no attempt to take into account the structure of the problem at hand.

Present-day convergence acceleration methods are mostly based on trying to achieve the optimum balance between speed of convergence and cost of iterations, and this is most often guided by studying the mathematical and/or physical structure of the particular problem at hand.

The empirical verification of Moore's Law (i.e. doubling of computational power at fixed cost every 18 months) over the last two decades has caused some to question the need for improved convergence

*ICASE, Mail Stop 403, NASA Langley Research Center, Hampton, Virginia 23681-2199, dimitri@icase.edu. This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-97046 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-2199.

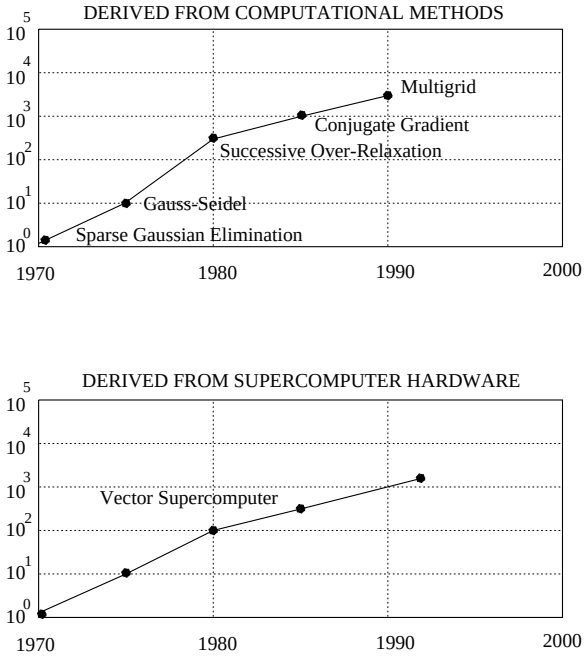

FIG. 1.1. *Illustration of advances due to algorithmic improvements and hardware improvements (reproduced from reference [1])*

acceleration techniques, opting instead to concentrate on incorporating additional physics through increased model complexity and/or resolution. Unfortunately, the incorporation of additional physics most often also increases the stiffness of the problem, resulting either in problems which simply cannot be solved by simple solution techniques, or take even longer to solve in spite of the availability of faster hardware. In the "Blue Book" report on scientific computing compiled by the National Science Foundation [1], a comparison of the enabling hardware advances versus the enabling algorithmic advances, reproduced here in Figure 1.1, serves to illustrate how the two fields have contributed almost equally to the overall advances in present-day simulation capability. The need for more efficient steady-state solution algorithms takes on even more significance when one considers the trend from steady-state Navier-Stokes solvers to unsteady solvers, and design optimization capabilities, which involve the solution of many intermediate steady-state or pseudo-steady-state problems for each run.

It is generally believed that there is the potential for one to two orders of magnitude improvement in convergence rates for steady-state Reynolds-averaged Navier-Stokes (RANS) calculations using refinements of current methods (e.g. fully converged solutions in hundreds of cycles rather than thousands or more cycles), while there exists the possibility for another one to two orders of magnitude improvement in devising radically new approaches to convergence acceleration [55] (e.g. textbook multigrid methods capable of convergence rates of 0.1). Many of these techniques are in their infancy, and remain to be applied to complicated problems.

1.2. Architectural Issues. Since reducing the overall cost of a solution is the ultimate goal of convergence acceleration, algorithms must be designed to run on cost effective hardware. The most often cited example is the need for algorithms to "parallelize" due to the advent of massively parallel computers. However, there are many other architectural issues which must be considered. While processor speed and memory

availability have increased exponentially over the last two decades, memory latency, the ratio of time required to fetch a variable from memory versus the time required for a floating-point operation has actually increased dramatically over this time period, as illustrated by the numbers in Table 1.1, reproduced here from reference [2]. While these numbers reflect single processor characteristics, these trends are exacerbated on massively parallel computer architectures due to the relatively higher latency and lower bandwidth of inter-processor communication. This has spawned the widespread use of hierarchical memory systems (such as multilevel caches) to hide latency. The development of latency tolerant algorithms, which usually implies the use of cache friendly strategies such as data re-use and locality, while minimizing inter-processor communication, is an important consideration in today's hardware environment.

An indispensable technique for improving cache re-use for unstructured mesh solvers involves data-reordering for locality, or sparse-matrix bandwidth minimization. Numerous re-ordering strategies have been developed [12, 14, 30], most often doubling or tripling the overall computational throughput. For parallel computations, mesh partitioning strategies which minimize the number of intersected mesh edges, and thus overall inter-processor communication, while maintaining load-balance, have also been the subject of extensive research [44, 18, 23]. While these techniques can be applied to most any solution algorithm, the development of latency tolerant solution algorithms, such as multigrid [36] or Newton-Krylov-Schwarz methods [9] is also an important consideration.

Similarly, one of the choices often confronting the algorithm researcher, particularly in the case of unstructured meshes, is the tradeoff between storage and computation. For example, in an implicit scheme, the original discretized equation set:

$$\frac{\partial w}{\partial t} + \mathbf{R}(w) = 0 \tag{1.1}$$

where w represents the solution vector, and $\mathbf{R}$ is the (non-linear) residual, is linearized about the current state, which yields the system:

$$[\frac{\mathbf{I}}{\Delta t} + \frac{\partial \mathbf{R}}{\partial w}]\Delta w = -\mathbf{R}(w) \tag{1.2}$$

The Jacobian matrix $\frac{\partial \mathbf{R}}{\partial w}$ represents a large sparse matrix. The storage required for the non-zeros of this matrix can easily be an order of magnitude larger than that required to assemble the residual $\mathbf{R}(w)$ in a purely explicit scheme.

An alternative to storing the entire Jacobian matrix is to compute the matrix vector product $\frac{\partial \mathbf{R}}{\partial w}\Delta w$ by finite difference as:

$$\frac{\partial \mathbf{R}}{\partial w}\Delta w = \mathbf{R}(w + \Delta w) - \mathbf{R}(w) \tag{1.3}$$

or, alternatively

$$\frac{\partial \mathbf{R}}{\partial w}\Delta w = \frac{\mathbf{R}(w + \epsilon\Delta w) - \mathbf{R}(w)}{\epsilon} \tag{1.4}$$

System	Memory latency [ns]	Clock speed [ns]	Ratio	FP ops per clock period	FP ops to cover memory latency
CDC 7600	275	27.5	10	1	10
CRAY 1	150	12.5	12	2	24
CRAY X-MP	120	8.5	14	2	28
SGI PowerChall	760	13.3	57	4	228
CRAY T3E-900	280	2.2	126	2	252

TABLE 1.1

Historical memory latencies (reproduced from: http://www.cray.com/products/systems/crayt3e/1200/performance.html)

where ϵ is a small parameter. This involves only the storage associated with the residual $\mathbf{R}(w)$, but involves multiple residual evaluations. For discretizations where the residual evaluation is an expensive operation (e.g. for example in combustion problems which may involve table lookups for reaction rates), storing the full Jacobian may be the most effective approach, while in memory limited cases, the second approach is more useful. Similar tradeoffs show up in many convergence acceleration algorithms. For example, in a Krylov technique such as GMRES, increasing the number of search directions can often be employed to enhance convergence at the expense of increased memory usage.

Clearly, the choice must not only depend on the algorithm, but on the *relative* cost of memory and cpu capability. If one assumes that the limiting factor for practical calculations in a production environment is the total runtime, then given the speed of the algorithm, one can deduce the maximum problem size which can be solved on the given hardware in the allotted time. If the memory requirements of this maximum problem size surpass the available memory, then the problem is said to be memory limited, and algorithms which avoid aggressive memory usage need to be employed. On the other hand, if the problem is not memory limited, then enhanced convergence acceleration or cpu time reduction may be sought using techniques which employ additional memory.

Assuming present day unstructured mesh steady state RANS algorithms are capable of converging of the order of 1 million points per processor within a 24 hour time period on modern day workstations or parallel machines, and assuming such machines can be outfitted at reasonable cost with 1 Gbyte of memory per processor, this results in a maximum memory usage of about 1 Kbyte (or 128 8-byte words) of memory per grid point. This is close to the amount required by a simple explicit scheme on unstructured grids. Therefore, steady-state unstructured grid RANS calculations are extremely memory limited in today's hardware environment. However, unsteady calculations, or design optimization calculations, which involve many equivalent steady-state calculations and thus much longer run times while using the same memory requirements, may in general be much less memory limited, and more memory intensive algorithms may be exploited. The algorithmic tradeoffs of memory versus cpu usage must therefore be guided by the nature of the problem to be solved, as well as the current economics of available hardware, which in itself is constantly in a state of change.

2. Classification of Convergence Acceleration Techniques.

2.1. Properties of the Jacobian. All convergence acceleration techniques rely in some form or other on the Jacobian matrix of the system of equations to be solved. The Jacobian is obtained as per equation (1.2), by linearizing the equations about the current state. The Jacobian represents the change of the residual

at a point with respect to the solution values. For unstructured meshes, the Jacobian consists of a large sparse matrix, the sparsity pattern of which depends on the stencil of the residual.

In sparse matrix terminology, the graph of a matrix is given by the set of edges formed by drawing a line between the row and column number associated with each non-zero entry in the sparse matrix. In the case of the Navier-Stokes equations, the Jacobian takes on a sparse block-matrix structure, where each non-zero entry belongs to a 5×5 submatrix. It is then useful to consider the graph of this sparse block-matrix as the set of edges joining row and column numbers identifying non-zero block sub-matrices, as depicted in Figure 2.1. These row and column numbers correspond to grid-point addresses. For a nearest neighbor stencil on tetrahedral meshes, the graph of the block matrix is identical to the graph of the grid, meaning that for every edge in the grid, there exists two block submatrices (upper and lower) in the Jacobian matrix. Including the non-zero block matrices on the diagonal of the Jacobian, the total number of non-zero entries in the Jacobian matrix becomes

$$5 \times 5 \times number\ of\ vertices\ + \tag{2.1}$$
$$5 \times 5 \times number\ of\ edges \times 2$$

In the case of tetrahedral meshes, where the number of edges is equivalent to 6 or 7 times the number of vertices, this results in a total of at least 325 non-zeros per grid point. This represents a considerable amount of storage over and above the 60 to 100 words per grid point that are generally required for an explicit solver on tetrahedral meshes [38, 42]. In general, only first-order discretizations result in nearest neighbor stencils. A typical second-order Jacobian involving distance-two neighbors requires of the order of 1400 words of storage per grid point.

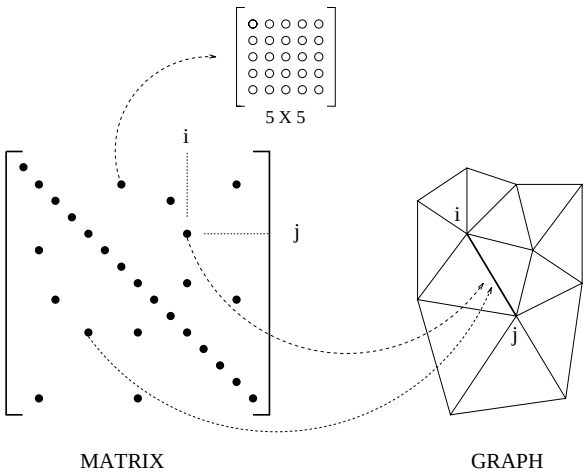

FIG. 2.1. *Illustration of graph of block-structured jacobian matrix arising from linearization of unstructured mesh discretization*

For these reasons, Jacobian matrices based on a first-order discretization are often employed in implicit schemes for solving equations based on second-order discretizations. Furthermore, in cases where the Jacobian matrix graph and the grid graph are equivalent, operations involving the Jacobian can be supported by an edge-based data-structure, similar to the one typically used for constructing the residual on unstructured meshes [38, 42, 31, 4].

On non-simplicial meshes, such as meshes involving prisms, pyramids and hexahedra, traditional finite-element discretizations result in non-nearest neighbor stencils, generally involving diagonally opposite points

within an element which are not connected by a mesh edge. In these cases, it is advantageous to reformulate the discretization into a nearest neighbor stencil when possible. For convective terms, this can be done using a finite-volume analogy, while for viscous terms, this is only possible under the assumption of incompressibility and constant viscosity using a finite difference approach, or by resorting to a thin-layer discretization [38, 16]. In cases where the discretization cannot be supported on a nearest-neighbor stencil, approximate Jacobians based on a reduced stencil are often employed.

When exact Jacobians are desired in the context of a non-nearest-neighbor stencil, a finite difference of the residual may be applied to approximate the Jacobian-vector product, a technique often employed in Newton-Krylov methods. For second-order schemes based on reconstruction, another approach consists of computing the second-order Jacobian-vector products using the first-order Jacobian operating on a reconstruction of the target vector as given by Barth [6]:

$$\left[\frac{\partial \mathbf{R}}{\partial w}\right]_2 \mathbf{p} = \left[\frac{\partial \mathbf{R}}{\partial w}\right]_1 \Psi(\mathbf{p}) \tag{2.2}$$

where the Jacobian subscript represents its order of accuracy, and $\mathbf{p}$ is an arbitrary vector, and $\Psi(\mathbf{p})$ represents the high-order reconstruction of $\mathbf{p}$ using the same reconstruction operator employed in the discrete residual operator $\mathbf{R}$.

2.2. Traditional Classification. For the purposes of this paper, the numerous convergence acceleration techniques may be classified into four groups: Direct Methods, Iterative Implicit Methods, Preconditioning Techniques, and Multigrid Methods. Such a classification is by no means complete or unambiguous, as many techniques can be shown to be equivalent to others, and most complete algorithms combine several techniques from more than one of these areas.

2.3. Direct Methods. Most implicit methods begin with the linearization of the governing equations:

$$[\frac{I}{\Delta t} + \frac{\partial \mathbf{R}}{\partial w}]\Delta w = -\mathbf{R}(w) \tag{2.3}$$

In a direct method, the augmented Jacobian $\left[\frac{I}{\Delta t} + \frac{\partial \mathbf{R}}{\partial w}\right]$ is inverted by Gaussian elimination. As the time-step is increased to infinity, a non-linear Newton's method is recovered. The quadratic convergence property of Newton's method results in very fast asymptotic convergence to machine precision [59]. However, the expense of the matrix inversion procedure, which scales as n^3, where n is the number of unknowns, is usually overwhelming, thus making direct methods non-competitive for large problems.

Rather than inverting the entire Jacobian matrix exactly, one may choose to divide the matrix into submatrices, through a partitioning process, and then to invert each submatrix directly. This substantially reduces the complexity of the inversion process, due to the smaller size of the number of unknowns n in each subdomain. The inversion of the global matrix can then be calculated from the inverted submatrices, and the (heretofore neglected) coupling between the domains. This is the basis for Shur-complement methods and Schwarz alternating methods, which specify particular treatments for the inter-domain coupling [50]. Although it is useful to think of these methods as performing direct inversions on the sub-matrices, in general any technique (often iterative) may be applied to achieve an approximation to the inverted submatrices. These methods are naturally suited for parallel computer architectures, where each submatrix may be associated with an individual processor.

2.4. Iterative Methods. Rather than inverting the Jacobian matrix directly at each time-step, the linear system resulting from equation (2.3) may be solved approximately at each time-step using an iterative method. This can substantially reduce the overall computational requirements since the linear system need not be solved to a high degree of accuracy at each time-step (i.e. it is only being used to advance to the next non-linear time-step), and because iterative methods generally exhibit lower computational complexity than direct inversion methods. A common practice consists of approximating the true Jacobian matrix in equation (2.3) with a Jacobian based on a linearization of a first-order discretization of the residual. This substantially reduces the memory requirements for schemes which explicitly store the Jacobian matrix, while resulting in a linear system that is more diagonally dominant and thus less stiff to solve. However, the mismatch between the discretization and Jacobian operators implies that the quadratic convergence property of Newton's method will not be attained.

Given a linear system to be solved:

$$\mathbf{A}x = b \tag{2.4}$$

a class of iterative schemes is obtained by splitting the matrix $\mathbf{A}$ into two parts $\mathbf{M}$ and $\mathbf{N}$

$$[\mathbf{M} + \mathbf{N}]\, x = b \tag{2.5}$$

The resulting iterative scheme is then defined as:

$$\mathbf{M}x^{k+1} = b - \mathbf{N}x^k \tag{2.6}$$

or equivalently

$$\mathbf{M}\left[x^{k+1} - x^k\right] = -\mathbf{A}x^k + b = -r^k \tag{2.7}$$

where r^k represents the residual for the linear system at the k^{th} step. Different choices of $\mathbf{M}$ lead to several well known interative schemes:

- $\mathbf{M} = \mathbf{I}$: Richardson's method
- $\mathbf{M} = \mathbf{D}$, where $\mathbf{D}$ is the diagonal : Jacobi iteration
- $\mathbf{M} = \mathbf{D} + \mathbf{E}$, where $\mathbf{D}$ is the diagonal and $\mathbf{E}$ is the lower triangular part of $\mathbf{A}$: Gauss Seidel Iteration
- $\mathbf{M} = \mathbf{D} + \mathbf{E}$ step, followed by $\mathbf{M} = \mathbf{D} + \mathbf{F}$ step, where $\mathbf{F}$ is the upper triangular part of $\mathbf{A}$: Symmetric Gauss Seidel

While the above splittings are applied directly to the Jacobian matrix, and thus depend on the ordering of the grid points, other splittings are possible which depend only on the formulation of the non-linear residual operator. For example, if $\mathbf{M}$ is taken as all the matrix entries which correspond to a set of grid lines (in a structured grid using a first-order linearization), a line-implicit scheme is obtained. A line-implicit scheme can be constructed for unstructured grids by grouping contiguous edges of the mesh together (using a graph algorithm) to form a set of lines, and then specifying $\mathbf{M}$ to be the entries corresponding to these edges in addition to the diagonal entries [17, 32, 34, 35]. Other types of splittings of the Jacobian matrix are possible, such as the convective eigenvalue-based splitting employed in the LU-SGS schemes[22, 52].

Krylov methods represent a different approach to iterative methods. The general idea is to obtain improved updates to the solution by using information generated at previous updates. There are a number of different Krylov methods which have been developed, but for CFD problems, the most prevalent Krylov

technique is the GMRES method [51]. Given the linear system of equation (2.4), GMRES(k) seeks updates of the form

$$x_k = x_0 + y_k \tag{2.8}$$

where x_0 is the initial guess, and y_k is the best possible correction over the Krylov subspace $K_m = span\left[r_0, Ar_0, A_2r_0, ..., A^{k-1}r_0\right]$ which minizes the residual:

$$||r_k|| = min_y||r_0 + Ay|| \tag{2.9}$$

where $r_0 = \mathbf{A}x_0 - b$ represents the initial residual. Thus, as the number of iterates increases, the Krylov subspace becomes larger, and the update approaches the exact result. In fact, a GMRES method converges exactly in n steps for a problem with n unknowns. However, since at each stage all previous solution vectors must be stored, and since the complexity of the minimization problem grows quadratically in the dimension of the Krylov subspace, it is generally only practical to use a small number of steps (i.e. $k \ll n$). In this case, GMRES must itself be applied iteratively, discarding all history effects after ($k \ll n$) cycles, and using the latest solution vector as the initial guess for the restarted GMRES procedure at the next iteration. An important aspect of GMRES is that the matrix $\mathbf{A}$ is never required explicitly, rather only matrix-vector products of the form $\mathbf{A}r$ are required. When the matrix $\mathbf{A}$ is a Jacobian of a non-linear residual, these matrix-vector products correspond to Frechet derivatives, which may be evaluated directly by finite-difference techniques.

2.5. Preconditioning. Preconditioning, while not a solution strategy in itself, is a technique which can be employed to accelerate existing solution methods. The principal idea, for a linear system, is to replace the original system described in equation (2.4) by the preconditioned system

$$\mathbf{PA}x = \mathbf{P}b \tag{2.10}$$

where $\mathbf{P}$ is a matrix which approximates $\mathbf{A}^{-1}$, while being simple to compute. (This corresponds to left preconditioning. A right preconditioned system is also possible [50], but will not be discussed here for brevity). Obviously the most effective preconditioner would be $\mathbf{A}^{-1}$ itself, but this would be too expensive to compute. The requirements for $\mathbf{P}$ are thus similar to those for the matrix $\mathbf{M}$ described above for iterative methods, and most iterative methods can be used as preconditioners. In fact, setting $\mathbf{P}^{-1} = \mathbf{M}$ and using a Richardson explicit method to solve the preconditioned system corresponds to using the iterative method defined by the original splitting of $\mathbf{A} = \mathbf{M} + \mathbf{N}$ (i.e. equations (2.6) and (2.7)). However, the power of preconditioned methods lies in their application to more complex iterative methods such multi-stage explicit or Krylov methods.

Choosing $\mathbf{P}^{-1} = \mathbf{D}$ corresponds to diagonal preconditioning, while choosing $\mathbf{P}^{-1}$ equal to the entries in the Jacobian which correspond to lines in the mesh (artificially constructed in the case of an unstructured mesh) corresponds to line or tridiagonal preconditioning [34, 35].

ILU preconditioners have been used with great success in CFD applications [60, 45, 6, 9, 66]. This class of preconditioners arises from an incomplete LU factorization of the Jacobian matrix. This factorization is incomplete in that any entries which arise during the factorization which fall outside of a pre-specified non-zero pattern are discarded. The most common approach, ILU(0), preserves the non-zero pattern of the original Jacobian matrix, although methods allowing additional fill-in (ILU(n) $n > 0$) have been employed occasionally [66]. If L and U represent the approximate factorizations, then the ILU preconditioner is defined

as $\mathbf{P} = (\mathbf{LU})^{-1}$. Note that ILU can also be used as an iterative method by setting $\mathbf{M}^{-1} = (\mathbf{LU})^{-1}$ in the definition for $\mathbf{M}$ above [60].

Preconditioning can also be employed for non-linear solution strategies, or in cases where the matrix $\mathbf{P}$ or $\mathbf{P}^{-1}$ cannot be expressed explicitly (e.g. consider the case where a multigrid method is used as a preconditioner). If a generic non-linear updating scheme is formulated as:

$$\Delta w = F[\mathbf{R}(w_1), \mathbf{R}(w_2), ..., \mathbf{R}(w_k)] \tag{2.11}$$

then the preconditioned scheme can be constructed simply by replacing the non-linear residuals in the above equation by the preconditioned residuals $\mathbf{PR}(w_1)..., \mathbf{PR}(w_k)$. In the case where the matrix $\mathbf{P}$ cannot be expressed explicitly, the preconditioned scheme is obtained by replacing the non-linear residuals in equation (2.11) by the corrections generated by the preconditioning scheme (i.e. multigrid scheme or other) applied to the corresponding solution vectors $w_1, ..., w_k$.

The term *local preconditioning* generally refers to preconditioners which depend only on values at the current grid point, with no influence from neighboring grid point values. The Jacobi preconditioner ($\mathbf{P^{-1}} = \mathbf{D}$) is an example of a local preconditioner. For the Navier-Stokes equations, the diagonal block $\mathbf{D}$ represents a 5×5 matrix at each grid point, which must be inverted [47, 40, 41, 43]. Many of the recently proposed local preconditioners [57, 29, 11, 63] involve changes to the discretization as well as preconditioning of the existing time-stepping or solution strategy. As will be discussed in the section on *current solution strategies*, these methods should not be classified simply as preconditioning techniques with regards to convergence acceleration, since traditional preconditioning methods have no influence on the accuracy of the final solution.

2.6. Multigrid Methods. The basic idea of a multigrid strategy is to accelerate the solution of a set of fine grid equations by computing corrections on a coarser grid. The motivation for this approach comes from an examination of the error of the numerical solution in the frequency domain. High-frequency errors, which involve local variations in the solution, are well annihilated by simple explicit methods. Low-frequency or more global errors are much more insensitive to the application of explicit methods. Multigrid strategies capitalize on this rapid initial error reduction property of explicit schemes. Typically, a multigrid scheme begins by eliminating the high-frequency errors associated with an initial solution on the fine grid, using an explicit scheme. Once this has been achieved, further fine grid iterations would result in a convergence degradation. Therefore, the solution is transferred to a coarser grid. On this grid, the low-frequency errors of the fine grid manifest themselves as high-frequency errors, and are thus eliminated efficiently using the same explicit scheme. The coarse-grid corrections computed in this manner are interpolated back to the fine grid in order to update the solution. This procedure can be applied recursively on a sequence of coarser and coarser grids, where each grid-level is responsible for eliminating a particular frequency bandwidth of errors.

Multigrid strategies are generally considered as convergence acceleration techniques, rather than solution methods themselves. In fact, they may be applied to any existing relaxation technique, explicit or implicit. Multigrid methods may be used to solve a linear system of equations, (c.f. equation (2.3)), or may be applied directly to the non-linear equations.

If the fine grid system of equations is written as

$$L_h w_h = f_h \tag{2.12}$$

where w_h is the solution which we seek, and the subscript h denotes fine grid values, and the residual r_h is defined as

$$L_h \overline{w}_h - f_h = r_h \tag{2.13}$$

where $\overline{w}_h$ represents the current estimate of the solution, then taking the difference of the above two equations yields

$$L_h w_h - L_h \overline{w}_h = -r_h \tag{2.14}$$

If the operator L_h is linear, the above equation may be reduced to an equation for the correction $\Delta w_h = w_h - \overline{w}_h$:

$$L_h \Delta w_h = -r_h \tag{2.15}$$

Assuming that the high-frequency errors in the solution have been eliminated by sufficient fine grid smoothing cycles, the remaining correction Δw_h which we seek must be smooth, and can therefore be computed more efficiently on a coarser grid by solving the equation

$$L_H \Delta w_H = -I_h^H r_h \tag{2.16}$$

where the subscript H denotes a coarser grid, and I_h^H is the *restriction operator* which interpolates residuals from the fine grid h to the coarse grid H. Once equation (2.16) is solved (typically recursively on coarser levels), the corrections are interpolated back to the fine grid as

$$\overline{w}_h^{new} = \overline{w}_h + I_H^h \Delta w_H \tag{2.17}$$

where I_H^h represents the *prolongation* operator which interpolates coarse grid corrections to the fine grid. Once these fine grid values have been updated, they may be smoothed again by additional fine grid iterations, and the entire procedure, which constitutes a single multigrid cycle, may be repeated until overall convergence is attained. This scheme, which is valid only for linear systems, is known as the multigrid correction scheme (CS).

In the case where L_h is a non-linear operator, the difference $L_h w_h - L_h \overline{w}_h$ in equation (2.14) can no longer be replaced by $L_h \Delta w_h$, and thus the above scheme must be modified. This is achieved by introducing a new coarse grid variable $\overline{w}_H$ defined as

$$\overline{w}_H = \overline{I}_h^H \overline{w}_h + \Delta w_H \tag{2.18}$$

where $\overline{I}_h^H$ represents an operator which interpolates fine grid solution variables to the coarse grid. The coarse grid equation equivalent to equation (2.16) can now be written as

$$L_H \overline{w}_H = -L_H \overline{I}_h^H \overline{w}_h - I_h^H r_h \tag{2.19}$$

which is solved for $\overline{w}_H$. The corrections (rather than the $\overline{w}_H$ variables themselves) are then interpolated back to the fine grid as

$$\overline{w}_h^{new} = \overline{w}_h^{old} + I_H^h (\overline{w}_H^{new} - \overline{I}_h^H \overline{w}_h^{old}) \tag{2.20}$$

This non-linear multigrid strategy is known as the Full Approximation Storage scheme (FAS). One of the principal advantages of this approach is that storage of the Jacobian matrix is avoided. This enables an efficient solution strategy which requires little more memory than that of the baseline (usually explicit) scheme employed to drive the smoothing process. This approach can be very beneficial especially for large unstructured mesh calculations.

Due to their structured grid heritage, multigrid methods have traditionally been developed from a geometric point of view. In this approach, coarse grids are constructed by removing alternate fine grid lines in each coordinate direction, and coordinate space interpolation routines (i.e. linear or bilinear) are used for restriction and prolongation. While this approach has been demonstrated successfully for unstructured grids [38, 42, 47, 28], recent interest has shifted towards algebraic or agglomeration multigrid methods. These approaches avoid the difficulties of constructing coarse meshes on complicated geometries associated with the geometric approach. Instead of consistent geometric coarse mesh levels, all that is required are coarse level matrices or graphs, which involve no geometric constraints, and can be constructed in a fully automatic manner. This is achieved using a graph algorithm, which identifies subsets of the fine grid points to be used as coarse grid points, or alternatively defines groupings of fine grid points which are comprised in coarse level control-volumes, as depicted in Figure 2.2.

In the purely algebraic multigrid algorithm, no underlying grid is defined, and only the matrix $\mathbf{A}$ of the linear system (c.f. equation (2.4)) is known. In this case the graph algorithm operates directly on the graph (i.e. non-zero entries) of the sparse matrix. The duality with geometric approaches arises from the fact that this matrix graph is equivalent to the graph of the grid in the case of a nearest neighbor stencil discretization on tetrahedral meshes. Once the coarse levels have been constructed, the coarse level equations to be solved are obtained by projecting the fine level operator as

$$L_H = I_h^H L_h I_H^h \tag{2.21}$$

Thus the coarse level equations are constructed in a completely algebraic manner, without any notion of spatial discretization. In many instances, simple piecewise constant operators are used for restriction and prolongation. In such cases, the above coarse grid operator reduces to summing constituent fine grid discrete equations to form the coarse level equation set [20, 25, 37, 33]. The particular fine grid equations to be summed for each coarse level equation are determined by the graph algorithm which constructs the coarse levels. For algebraic multigrid methods, which are only valid for linear systems, this corresponds to the direct summation of non-zero entries in the fine level matrix. Agglomeration multigrid represents an extension of these ideas to non-linear systems. In this approach, only the solution independent terms are summed in going from fine to coarse levels, while the solution dependent terms are obtained from coarse level variables interpolated up from the fine level solution, following the FAS approach.

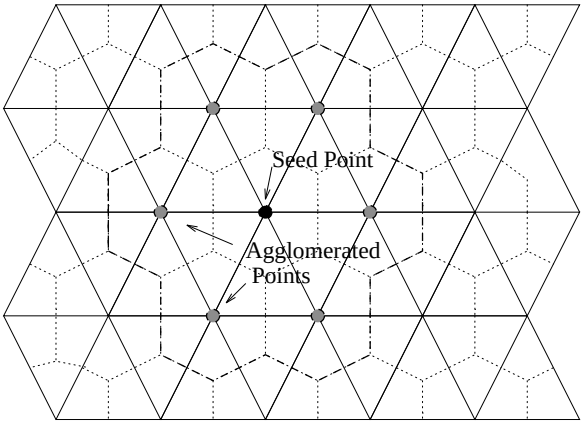

FIG. 2.2. *Illustration of Agglomeration of Fine Grid Vertices into Coarse Level Groupings or Control Volumes*

2.7. Defect-Correction Schemes. Defect-correction schemes enable the solution of a higher order discretization using a solution strategy based on a lower order discretization. Similarly to preconditioners and multigrid methods, defect-correction schemes represent a convergence acceleration technique, which can be applied to existing solution methods.

Assuming we want to solve the higher-order non-linear system $\mathbf{R_{high}}(w) = 0$, and have an efficient technique for solving the lower order system $\mathbf{R_{low}}(w) = 0$, a defect-correction iteration iteration can be formulated as:

$$\mathbf{R_{low}}(w^{n+1}) = \mathbf{R_{low}}(w^n) - \mathbf{R_{high}}(w^n) \tag{2.22}$$

where the right-hand-side represents the defect, or difference between the higher and lower-order operator. This iteration achieves convergence when $\mathbf{R_{high}}(w) = 0$, since this is the situation in which the iteration scheme defined by equation (2.22) produces no additional changes in w. Defect-correction strategies can be applied directly to non-linear operators without the need for forming and storing a Jacobian. They are often employed in conjunction with multigrid techniques, using a first-order multigrid strategy to solve a second-order discretization [26]. This enables the use of first-order multigrid methods which are generally very efficient and robust. In the algebraic or agglomeration multigrid case, the use of a first-order method may be the enabling factor for preserving the correspondence between the matrix graph and the grid graph.

A common practice in multigrid implementations is to employ a first-order discretization on the coarse level grids, while retaining the second-order accurate discretization on the fine grid. This corresponds to the use of a defect-correction scheme on the coarser levels of the multigrid algorithm [21, 38, 43].

3. Linear and Non-Linear Methods. One of the principal choices in designing a solution strategy for unstructured mesh CFD problems concerns the treatment of the non-linear behavior of the equations. This is an important consideration, since the resulting memory and cpu requirements can vary significantly between the linearized and the non-linear approaches. The purpose of the following discussion is to demonstrate the equivalence between several well known linear and non-linear strategies.

The essence of the argument is that all non-linear approaches can be shown to be equivalent to a linearized approach where Jacobian-vector products are replaced by a finite difference of the residual as:

$$\frac{\partial \mathbf{R}}{\partial w}\Delta w \approx \mathbf{R}(w + \Delta w) - \mathbf{R}(w) \tag{3.1}$$

or, alternatively

$$\frac{\partial \mathbf{R}}{\partial w} r \approx \frac{\mathbf{R}(w + \epsilon r) - \mathbf{R}(w)}{\epsilon} \tag{3.2}$$

where r is an arbitrary vector, and ϵ is a small parameter. This is the basis for Newton-Krylov methods, which use a sequence of the above finite-difference evaluations to approximate the Jacobian-vector products $\mathbf{A}r$ required by Krylov methods such as GMRES (c.f. equation (2.9)). Other solution strategies, such as preconditioned non-linear iteration methods, defect-correction approaches, and the Full Approximation Storage (FAS) multigrid method can also be recast as modifications of the equivalent linearized approach using finite differencing of the non-linear residual operator.

Equation (2.3) constitutes the starting point for all linear methods. Neglecting the contribution of the time-step (valid for large time-steps) the linearized system of equations becomes

$$\frac{\partial \mathbf{R}}{\partial w} \Delta w = -\mathbf{R}(w^n) \tag{3.3}$$

A class of iterative schemes is defined by splitting the Jacobian matrix as $\frac{\partial \mathbf{R}}{\partial w} = \mathbf{M} + \mathbf{N}$ (c.f. equation (2.5)), thus producing the following iterative scheme:

$$[\mathbf{M}] \, {\Delta w_l}^{k+1} = -\mathbf{R}(w^0) - [\mathbf{N}] \, {\Delta w_l}^k \tag{3.4}$$

which can be rewritten as

$$[\mathbf{M}] \left[{\Delta w_l}^{k+1} - {\Delta w_l}^k \right] = -\mathbf{R}(w^0) - [\mathbf{M} + \mathbf{N}] \, {\Delta w_l}^k \tag{3.5}$$

where Δw_l represents the correction generated by this linear iteration strategy, and the superscript refers to the linear iteration count. The corresponding non-linear preconditioned iteration strategy can be written as

$$[\mathbf{M}] \, {\Delta w_{nl}}^{n+1} = -\mathbf{R}(w^n) \tag{3.6}$$

where $w^{n+1} = w^n + {\Delta w_{nl}}^{n+1}$, and the nl subscript refers to the non-linear update. In this case, the residual operator $\mathbf{R}(w^n)$ is updated (and therefore reevaluated) at each iteration in the non-linear scheme, whereas it is held fixed in the linear approach. On the other hand, the linear approach requires storage for all Jacobian terms $[\mathbf{M} + \mathbf{N}]$, while the non-linear approach only requires storage for the preconditioning matrix $[\mathbf{M}]$. At convergence the non-linear updates ${\Delta w_{nl}}^n$ vanish, while the linear updates ${\Delta w_l}^k$ approach a constant value. The correspondence between the linear and non-linear updates is given by:

$${\Delta w_{nl}}^{k+1} = {\Delta w_l}^{k+1} - {\Delta w_l}^k \tag{3.7}$$

which by summation yields

$${\Delta w_l}^k = \sum_{n=0}^{k} {\Delta w_{nl}}^n \tag{3.8}$$

with the assumption ${\Delta w_l}^0 = 0$. Substituting these expressions into equation (3.5) produces the expression

$$[\mathbf{M}] \, {\Delta w_{nl}}^{k+1} = -\mathbf{R}(w^0) - [\mathbf{M} + \mathbf{N}] \sum_{n=0}^{k} {\Delta w_{nl}}^n \tag{3.9}$$

which is equivalent to equation (3.6) provided

$$[\mathbf{M}+\mathbf{N}]\sum_{n=0}^{k}\Delta {w_{nl}}^{n} \approx \mathbf{R}(w^0)-\mathbf{R}(w^k) \tag{3.10}$$

which corresponds to approximating the Jacobian vector product by a finite-difference approximation, assuming the preconditioning matrix $[\mathbf{M}]$ to be constant. In the case where the residual is linear in w, the two schemes are exactly equivalent. Local preconditioning methods, such as Jacobi preconditioning employed with explicit time-stepping, are thus equivalent to (block)-Jacobi iterations applied to the linearized system under these assumptions. The common practice of using a first-order Jacobian in the linearization (c.f. equation (3.3)), means that the approximation in equation (3.10) will be less accurate, thus contributing to further differences between the non-linear and linear (defect-correction) iteration strategies.

In the case of the defect-correction scheme, (defined by equation (2.22)), using the finite-difference approximation to the low-order residual operator

$$\frac{\partial \mathbf{R_{low}}}{\partial w}\Delta w = \mathbf{R_{low}}(w^{n+1})-\mathbf{R_{low}}(w^n) \tag{3.11}$$

enables the scheme to be rewritten as

$$\frac{\partial \mathbf{R_{low}}}{\partial w}\Delta w = -\mathbf{R_{high}}(w^n) \tag{3.12}$$

Thus, the popular strategy of employing a Jacobian derived from a first-order discretization operator in many linearized implicit schemes (often dictated by memory requirements) corresponds to a defect-correction scheme. Note that this linearized defect-correction scheme can also be interpreted as a preconditioning scheme, where the preconditioning matrix is given by $\mathbf{P}=\left[\frac{\partial \mathbf{R_{low}}}{\partial w}\right]^{-1}$.

The FAS multigrid scheme applied directly to the non-linear equations can also be shown to be equivalent to a linear multigrid correction scheme (CS) applied to the linearized form of the equations. In this latter case, equation (3.3) is solved using the multigrid correction scheme, and the operator L previously used to define the multigrid scheme corresponds to the Jacobian $\frac{\partial \mathbf{R}}{\partial w}$. The coarse grid equations (c.f. equation (2.16)) thus become

$$\left[\frac{\partial \mathbf{R}}{\partial w}\right]_H \Delta w_H = -I_h^H \mathbf{R_h} \tag{3.13}$$

using H and h to denote the coarse and fine grids respectively. Replacing the left-hand side of this equation with a finite difference of the residual yields

$$\mathbf{R_H}({w_H}^{n+1})-\mathbf{R_H}({w_H}^{n}) = -I_h^H \mathbf{R_h} \tag{3.14}$$

which is equivalent to the FAS coarse grid equations described in equation (2.19), since ${w_H}^n$ represents the values interpolated up from the fine grid, and the operator L used to define the FAS scheme now corresponds to the non-linear residual operator $\mathbf{R}$. This also provides the justification for interpolating the change ${w_H}^{n+1}-{w_H}^n$ back to the fine grid rather than the variables ${w_H}^{n+1}$ themselves.

The correspondence between linear and non-linear schemes demonstrated above has implications for the design of efficient solution strategies. While substantial differences between the linear and non-linear approach may be expected in the initial phases of convergence when large transients are present, in the asymptotic convergence region, where the updates become very small and the linearization becomes more

accurate, the two approaches can be expected to behave almost identically. Solution strategies which provide a good balance between memory and cpu usage, as discussed in the introduction, can therefore be designed by making the most appropriate decisions concerning the treatment of the governing equation non-linearities.

4. Current Solution Strategies.

4.1. Block Matrices and Local Preconditioners. One of the most prevalent techniques in modern-day solution algorithms is the treatment of systems of equations, such as the Navier-Stokes equations, as fully coupled systems at each grid point. This involves writing the Jacobian matrix as a block-matrix, and performing all matrix operations at the block level (i.e. block diagonal inversions, block tridiagonal, block ILU etc.)

It has been well established that simple scalar methods (such as local time-stepping) can be improved upon by replacing them with methods that take into account point-wise coupling, such as Jacobi preconditioning techniques [47, 40, 41, 43]. For the laminar Navier-Stokes equations, the block sub-matrices are 5×5 matrices, which can be directly inverted or factorized at reasonable cost.

For Reynolds-averaged Navier-Stokes computations, the turbulence model and flow equations are often solved in an uncoupled fashion [34, 35], although full point-wise coupling of these equations has been suggested more recently [58].

Point-wise coupling can be added to schemes which lack this property through local preconditioning techniques. For example, a Jacobi preconditioner can be used to modify the simple explicit scheme

$$\frac{\Delta w}{\Delta t} = -\mathbf{R}(w) \tag{4.1}$$

as

$$[\mathbf{D}]\,\Delta w = -\mathbf{R}(w) \tag{4.2}$$

where the diagonal time-step matrix $\frac{[\mathbf{I}]}{\Delta t}$ has been replaced by the block diagonal $[\mathbf{D}]$ of the Jacobian matrix.

In fact, most so-called "local preconditioners" [57, 29, 11, 63] are more sophisticated than this simple example. These methods have been devised to reduce the disparity in eigenvalues of stiff systems, as in the case of nearly incompressible flow, where there exists a large disparity between the acoustic and pressure modes in the Euler or Navier-Stokes equations. This is achieved by modifying the discretization as well as preconditioning the time-stepping procedure. If the discrete residual is written as

$$-\mathbf{R}(w) = \sum_{k=1}^{neighbors} \frac{1}{2}(\mathbf{F}(w_i) + \mathbf{F}(w_k)).\mathbf{n}_{ik} - \frac{1}{2}|\mathbf{A}_{ik}|(w_k - w_i) \tag{4.3}$$

where $\mathbf{F}$ represents the convective fluxes, $\mathbf{n}_{ik}$ the outwards normal at a control-volume interface, $\mathbf{A}$ the dissipation or Roe matrix arising from an approximate Rieman solver, and the summation is over all neighboring vertices k of vertex i, then the application of a typical local preconditioner can be written as

$$\left[\overline{\mathbf{P}}_i\right] \Delta w_i = \sum_{k=1}^{neighbors} \frac{1}{2}(\mathbf{F}(w_i) + \mathbf{F}(w_k)).\mathbf{n}_{ik} - \frac{1}{2}\mathbf{P}|\mathbf{P}^{-\mathbf{1}}\mathbf{A}_{ik}|(w_k - w_i) \tag{4.4}$$

where the dissipation terms in the residual have been modified by the "preconditioning" matrix $[\mathbf{P}]$, and the time-stepping history by the "preconditioning" matrix $[\overline{\mathbf{P}}]$.

An alternative interpretation of these methods arises if one simply considers the Jacobi preconditioner described by equation (4.2), applied to the modified residual of equation (4.4) which, through a linearization of the new residual terms can be written as [34]:

$$(\sum_{k=1}^{neighbors} \frac{1}{2}\mathbf{P}|\mathbf{P^{-1}}\mathbf{A}_{ik}|)\Delta w_i = \sum_{k=1}^{neighbors} \frac{1}{2}(\mathbf{F}(w_i)+\mathbf{F}(w_k)).\mathbf{n}_{ik} - \frac{1}{2}\mathbf{P}|\mathbf{P^{-1}}\mathbf{A}_{ik}|(w_k - w_i) \tag{4.5}$$

In this case, the $\left[\overline{\mathbf{P}}_i\right] = (\sum_{k=1}^{neighbors} \frac{1}{2}\mathbf{P}|\mathbf{P^{-1}}\mathbf{A}_{ik}|$ matrix is completely determined by the modifications to the discrete residual (i.e. the $[\mathbf{P}]$ matrix). In fact, in the case where the $\left[\mathbf{P}^{-1}\mathbf{A}\right]$ matrix is diagonal (i.e. scalar dissipation), the $\left[\overline{\mathbf{P}}\right]$ and $[\mathbf{P}]$ matrices become identical (to within a multiplicative constant, neglecting the differences between $\mathbf{P}$ evaluated at the interfaces ik and evaluated directly at the vertex i). This offers a different view-point of the mechanism involved for relieving low Mach number stiffness in compressible flow formulations. Rather than a technique which attempts to alter the time history of the convergence process, this type of "preconditioning" may be thought of as one which simply alters the discretization of an implicit (Jacobi) scheme to yield a more consistent and less stiff system of equations.

4.2. Linear Implicit Methods based on First-Order Jacobians. One of the most common techniques in the context of linear implicit methods consists of using an approximate Jacobian obtained through a linearization of a first-order discretization. This amounts to a linearized defect-correction scheme, as demonstrated by equation (3.12), therefore retaining full second-order spatial accuracy at convergence. The advantages include lower storage requirements for the simpler Jacobian, more diagonally dominant linear systems, and often a Jacobian matrix graph which corresponds to the grid graph, thus enabling Jacobian assembly and manipulation using edge-based data-structures. In all cases, the Jacobian matrices are treated as block matrices.

Jacobi, Gauss-Seidel, or SSOR linear solvers have been developed in the literature for unstructured meshes [64, 7, 3]. In general, these solvers perform reasonably well for simple problems, but degrade as the problem size or stiffness increases [60]. For these reasons, the schemes are often used as preconditioners for Krylov iteration methods, or as smoothers for multigrid. In practice, the ILU(0) preconditioned GMRES linear solver has consistently been found to offer the best combination of robustness and efficiency [60, 61], while simpler schemes such as Gauss-Seidel have been used as effective multigrid smoothers [45]. Figure 4.1, reproduced here from reference [45] illustrates the convergence rates obtained using a red-black Gauss-Seidel iterative scheme and an ILU(0) pre-conditioned GMRES iterative scheme, both used as solvers and as smoothers within a non-linear multigrid strategy. In all cases, the iterative solvers are used to converge the linear system corresponding to the defect-correction scheme (i.e. a Jacobian derived from a first-order discretization). The problem being solved consists of two-dimensional inviscid flow over a 4-element airfoil on an unstructured grid of 25,862 vertices. The Gauss-Seidel scheme used as a solver yields the slowest convergence, while the stronger ILU pre-conditioned GMRES iterative solver achieves substantially faster convergence. However, both schemes produce superior convergence rates when used as multigrid smoothers. These comparisons are made based on the number of outer non-linear iterations. Figure 4.2 provides a comparison in terms of CPU time, which illustrates the cost of an ILU-GMRES iteration over a Gauss-Seidel iteration. In this case, the most efficient solver is the Gauss-Seidel multigrid scheme using a W-cycle. In fact, even the Gauss-Seidel based multigrid V-cycle is more efficient than the equivalent ILU-GMRES multigrid

V-cycle, due to the greater expense of the stronger ILU-GMRES smoother. In this particular case, the stronger smoother does not produce a large enough increase in convergence speed to cover its extra expense. However, in more stiff problems, such as viscous flow simulations, the smoothing effectiveness of the simple Gauss-Seidel scheme may degrade faster than that of the ILU-GMRES scheme, making the latter a more effective solution scheme in terms of CPU time.

In cases where the Jacobian is stored explicitly, the memory requirements are significantly higher than those required by an explicit or FAS multigrid scheme. Recomputing the Jacobian terms "on-the-fly" as in the LU-SGS scheme [22, 52] can be used to reduce memory usage. This can also be achieved with the GMRES solver, using finite-differences to approximate the Jacobian-vector products. However, matrix-based preconditioners such ILU, which are instrumental in the success of GMRES methods, still require storage levels comparable to that required by the Jacobian matrix.

While good convergence rates have been demonstrated with linearized defect-correction schemes, recent analysis (based on periodic boundary conditions) indicates that the spectral radius of such schemes tends to unity as the grid is refined [56]. While the level of grid resolution at which this effect becomes noticeable has yet to be determined, this suggests that defect-correction schemes may be best suited as smoothers or preconditioners for an outer non-linear Newton or multigrid scheme.

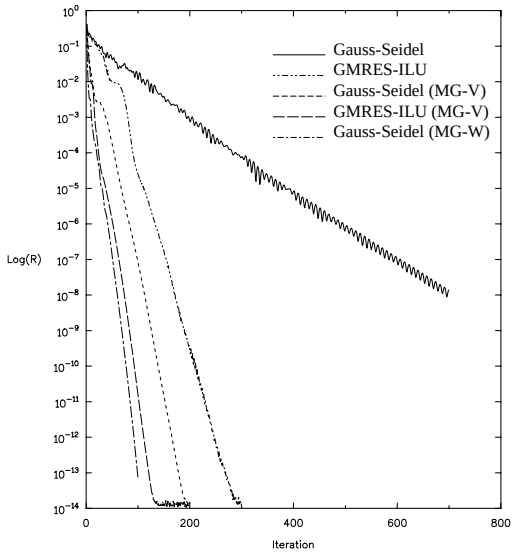

FIG. 4.1. *Comparison of convergence rates in terms of non-linear iteration counts achieved by Gauss-Seidel and ILU-GMRES used as solvers and as smoothers for multigrid for inviscid flow over four-element airfoil configuration (reproduced from [45])*

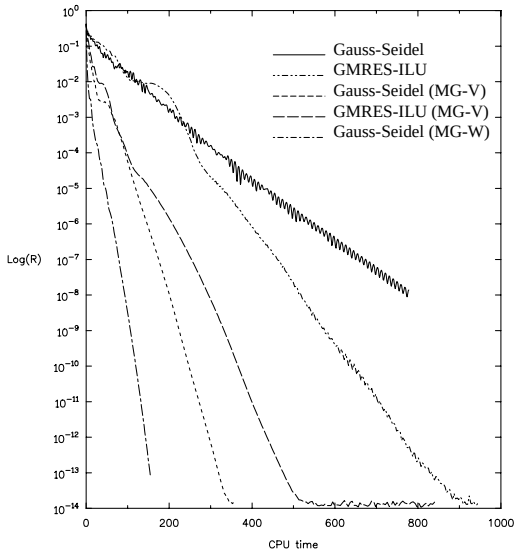

FIG. 4.2. *Comparison of convergence rates in terms of CPU time achieved by Gauss-Seidel and ILU-GMRES used as solvers and as smoothers for multigrid for inviscid flow over four-element airfoil configuration (reproduced from [45])*

4.3. Linear Implicit Methods based on the Full Jacobian. Since the Jacobian of a full second-order discretization contains a much larger proportion of non-zeros than that based on a first-order Jacobian, this more complex Jacobian is seldom assembled and stored explicitly. For second-order discretizations based on higher-order reconstruction, Barth [6] has shown how the higher-order Jacobian matrix-vector product may be expressed as a product of a first-order Jacobian matrix applied to a reconstructed vector, as per equation (2.2). This technique, which avoids storing the full second-order Jacobian, has been applied successfully in the context of a fully implicit ILU(0) pre-conditioned GMRES scheme [6, 61]. The more common practice of evaluating the second-order Jacobian-vector product by finite-differences has seen widespread use in Newton-Krylov methods, particularly employing ILU preconditioned GMRES solvers. Although storage of the Jacobian is avoided in this manner, the ILU preconditioners can require substantial amounts of storage. In most cases an ILU(0) preconditioner based on a factorization of a first-order Jacobian is employed [60, 45, 9, 24, 66], although faster convergence with additional memory overhead has been demonstrated for ILU(1) and ILU(2) preconditioners on structured meshes [66]. Barth [5] has demonstrated increased convergence rates using an ILU(0) factorization of the full second-order Jacobian.

Newton-Krylov methods have become relatively robust and efficient, and parallelize efficiently when the preconditioning step is applied locally to partitioned subdomains, and the Schwarz procedure is employed to account for the coupling between the domains [6, 9, 10]. If the linear system is solved to adequate tolerance at each non-linear step, quadratic convergence is generally observed, providing very rapid convergence to machine zero in terms of the number of non-linear updates. This behavior is observed in Figure 4.3, reproduced from [24], where the convergence history achieved by a Newton-Krylov-Schwarz method running in parallel on a CRAY T3E is displayed. The problem being solved consists of inviscid flow over an ONERA M6 wing, on a relatively fine grid of 2.7 million vertices. This calculation achieved an aggregate computational rate of 32 Gflops on 512 processors of the CRAY T3E, demonstrating the suitability of such algorithms for parallel machines. When increased numbers of domains are used, global convergence of the Newton-Krylov-Schwarz method degrades only slightly, as depicted in the figure. In spite of their rapid asymptotic convergence,

one of the drawbacks of Newton-Krylov methods, is the relatively slow convergence of these methods in the inital phases of the calculation. The use of continuation techniques (i.e. such as grid sequencing) is currently under investigation to accelerate this phase of the calculations [61].

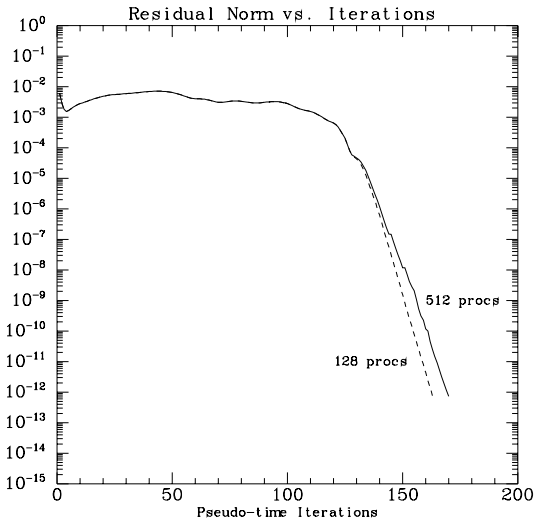

FIG. 4.3. *Illustration of quadratic convergence behavior of Newton-Krylov-Schwarz method for inviscid three-dimensional flow on 2.7 million point grid running in parallel on the CRAY T3E and effect of number of domains on convergence (reproduced from [24])*

4.4. Unstructured Multigrid Methods. Most early multigrid methods for unstructured grids were based on the geometric approach [38, 42, 47, 28]. In this approach, coarse level grids are constructed either manually using a grid generator, or using an automatic point-removal process followed by retriangulation [15]. Restriction and prolongation operators are constructed using linear interpolation between the coarse and fine grids of the multigrid sequence. Figure 4.4 illustrates a sequence of grids employed by a geometric multigrid algorithm for computing the inviscid flow over an aircraft configuration. The finest grid contains a total of approximately 804,000 vertices, and the flow conditions (Mach = 0.77, incidence = 1.116 degrees) result in transonic flow. Figure 4.5 illustrates the speedup achieved by the multigrid algorithm over the single grid explicit scheme for this case. A residual reduction of roughly six orders of magnitude in 100 cycles is observed for the multigrid algorithm, which is an order of magnitude faster than the single grid explicit scheme. While the success of these methods in reducing solution time is undeniable, the difficulties associated with constructing consistent coarse grid levels, particularly in three dimensions, was soon found to be a practical limiting factor.

For this reason, algebraic multigrid methods have been developed, particularly in the commercial software field, were usability is of the utmost importance [19, 46, 62]. Since algebraic multigrid schemes are linear methods, they must be applied to the linearized equations (c.f. equation (2.3) or (3.3)). For example, the additive correction method [19, 33] corresponds to an algebraic multigrid method which has been applied to a first-order Jacobian linearization of the governing equations. A similar approach has been described in reference [62] using an algebraic multigrid method operating on a first-order Jacobian linearization, using a locally preconditioned multi-stage explicit smoother.

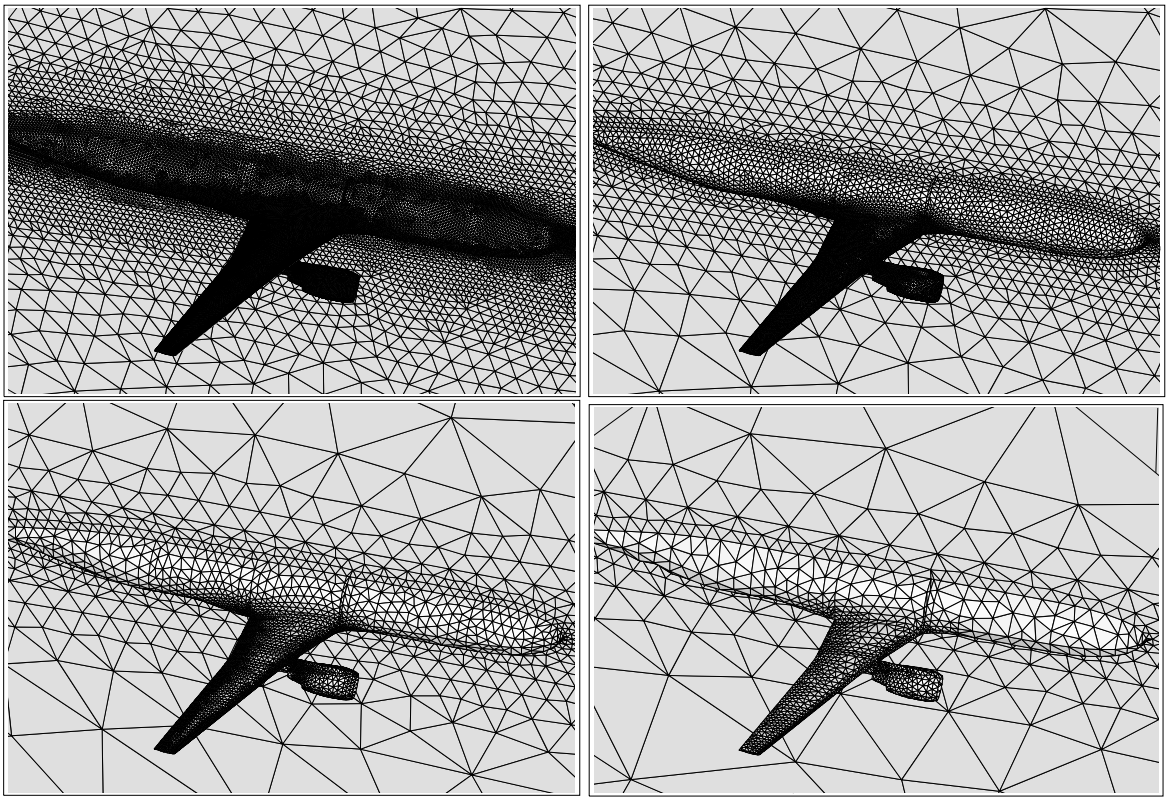

FIG. 4.4. *Sequence of Geometric Grids Employed for Geometric Multigrid for Computation of Flow over Aircraft Configuration*

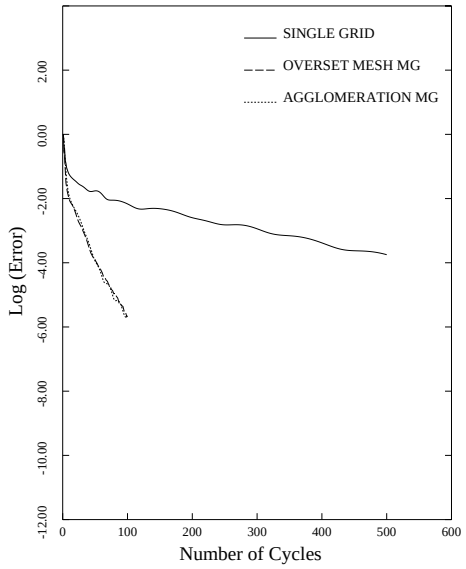

FIG. 4.5. *Comparison of convergence rate achieved by geometric and agglomeration multigrid methods with single grid explicit scheme for inviscid transonic flow over aircraft configuration*

One of the difficulties associated with algebraic multigrid methods applied to linear systems is the large memory overheads they incur. Recall that the storage of a first-order Jacobian requires four to five times more memory than a purely explicit scheme. Agglomeration multigrid methods [27, 53, 37, 33] combine the automation of algebraic multigrid methods with the low-memory overheads associated with FAS multigrid methods. In this approach, non-geometric coarse levels (i.e. grouping of fine grid vertices) are constructed using a graph algorithm, and non-linear coarse level equations are formed using algebraically constructed stencil coefficients and flow variables interpolated from up the fine levels.

Figure 4.6 depicts the graph corresponding to the fine grid of Figure 4.4, and three additional coarser graphs generated and employed by the agglomeration multigrid scheme to recompute the same case described above in the context of geometric multigrid. The convergence rate of the agglomeration multigrid is compared with the geometric multigrid result in Figure 4.5, where the two convergence rates are seen to be almost identical.

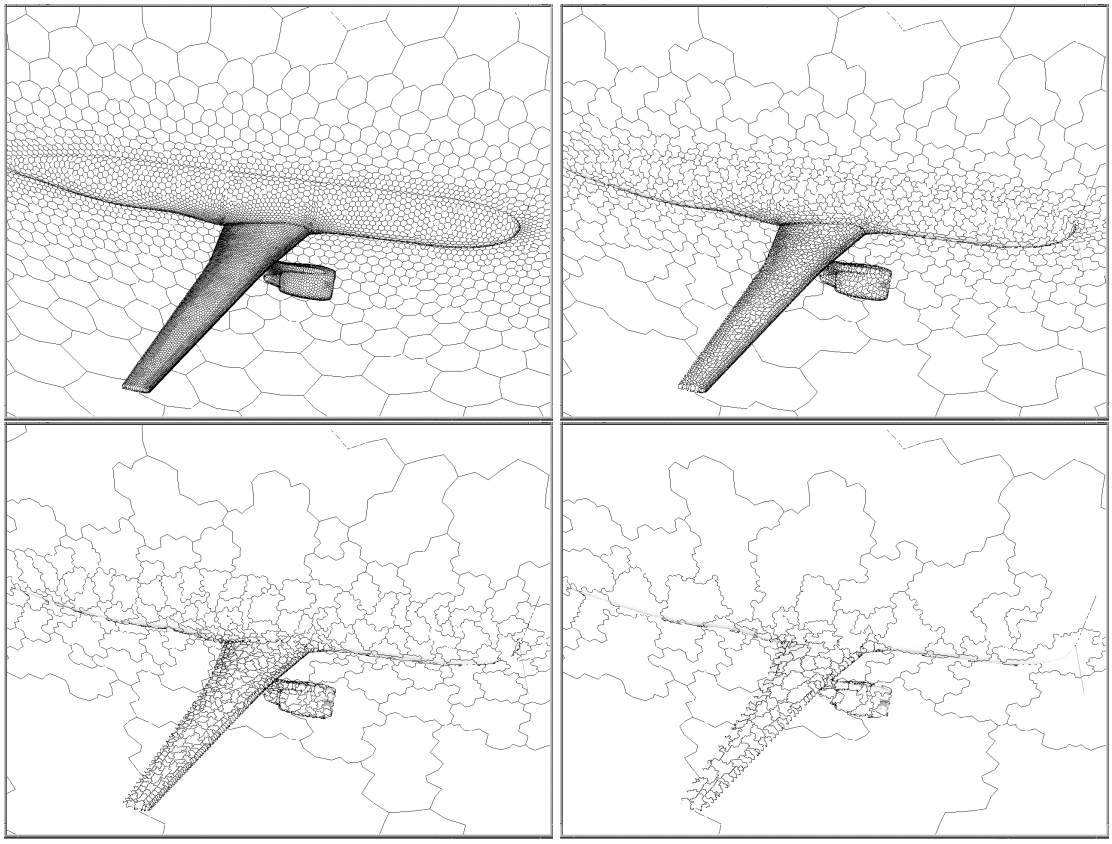

FIG. 4.6. *Fine and coarse level agglomerated graphs employed by the agglomeration multigrid algorithm for computation of inviscid transonic flow over aircraft configuration*

Agglomeration multigrid methods have also been combined with local preconditioners to produce enhanced convergence rates with minimal additional memory overheads [34, 35]. Additional convergence acceleration can be obtained by using multigrid methods as preconditioners for Krylov iteration techniques such as GMRES [65, 41, 34] In this case, the GMRES method requires additional storage, which scales linearly with the prescribed number of search directions in the GMRES algorithm. The number of search directions must therefore be determined by considering the balance between increased memory usage and enhanced convergence acceleration produced by the overall method. In most instances, a maximum of 20 to 30 search

directions are employed in GMRES.

Figure 4.7 (reproduced from [34]) compares the convergence rates achieved by various multigrid strategies for the computation of two-dimensional viscous turbulent flow over a three-element airfoil configuration on a grid of 30,562 vertices, at a Mach number of 0.2, an incidence of 16 degrees, and a Reynolds number of 5 million. The original multigrid scheme converges rather slowly compared to the rates displayed for the inviscid flow solutions in Figure 4.5. The directional-implicit multigrid method, to be described in the following section, achieves only modest speedup over the original algorithm in this case.

However, the addition of local preconditioning (designed to relieve stiffness associated with low-Mach number effects), as described by equation (4.5), results in substantially improved convergence for this case, most likely because of the large regions of low Mach
number flow which are present in the solution. Finally, the use of the entire pre-conditioned multigrid solution strategy as a preconditioner for GMRES results in even faster convergence, driving the residuals to machine zero in just over 400 equivalent multigrid cycles.

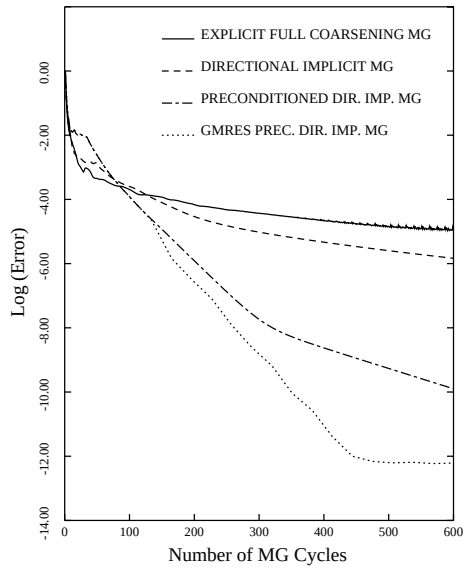

FIG. 4.7. *Comparison of various multigrid strategies for computation of two-dimensional viscous turbulent flow over three-element airfoil on grid of 30562 vertices (reproduced from [34])*

4.5. Anisotropic Problems. For high Reynolds number viscous flow simulations, highly stretched grids are required for efficiently resolving the thin boundary-layer and wake regions. These grids can often exhibit aspect ratios of the order of 10^4, thus resulting in very stiff systems of discrete equations. Much recent work has been devoted towards developing efficient solution strategies for highly anisotropic problems.

Unstructured grid based solution strategies are at a distinct advantage with regards to highly anisotropic problems. Due to the lack of a global grid structure, anisotropic regions may be identified and localized, thus enabling the development and application of adaptive solution strategies, which can be designed to take into account the local degree of anisotropy. For example, semi-coarsening techniques often used in structured multigrid methods, where grid points are removed in only one coordinate direction, cannot deal efficiently with grids containing conflicting stretching directions. For unstructured mesh multigrid methods, semi-coarsening methods generalize to directional-coarsening methods, which can produce an optimal coarsening

based on the local degree of anisotropy in the mesh.

While regular (fully coarsened) multigrid methods degrade rapidly with increasing grid stretching, strongly implicit methods may be more resilient with regards to anisotropic problems. This is due to the fact that the implicitness required to relieve the anisotropy induced stiffness may inherently reside in the existing solution strategy. For example, a direct method would presumably be completely insensitive to the degree of anisotropy in a problem. In most cases however, implicit solution strategies designed for isotropic problems degrade with increasing anisotropy. This indicates the need for research into anisotropy-capable implicit methods.

One approach consists of forming local line structures in the unstructured grid, by joining together strongly connected neighboring points, based on the local degree of anisotropy in the mesh, and solving the reduced implicit system on these lines [34, 35]. As an example, the application of a weighted-graph algorithm [35] which identifies and groups together edges joining strongly connected neighbors results in the set of lines depicted in Figure 4.8 when applied to the stretched two-dimensional grid shown in the same figure.

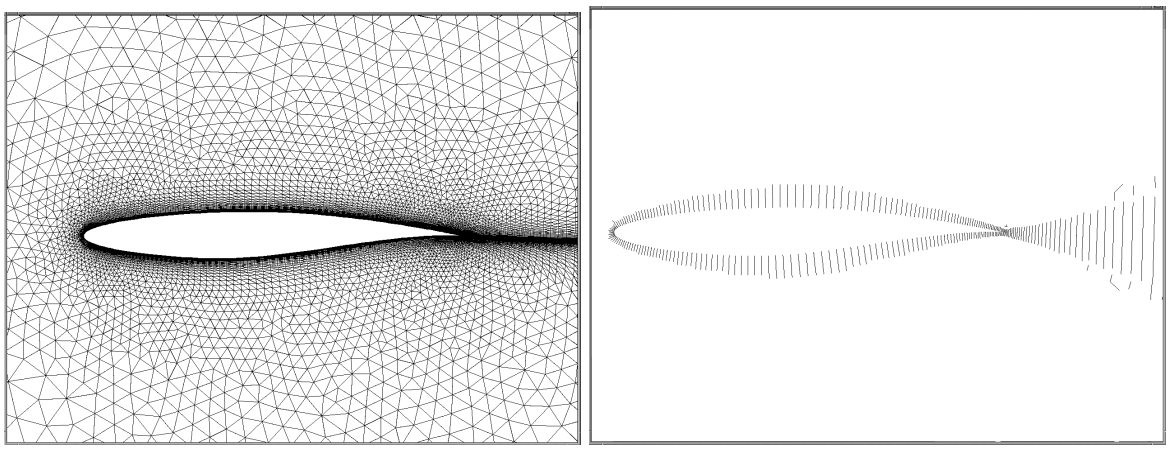

FIG. 4.8. *Initial stretched unstructured grid and extracted set of lines using weighted-graph algorithm for line-implicit multigrid smoother (reproduced from [35])*

Another approach consists of partitioning the domains, within which strongly implicit solution procedures are applied, such that the degree of coupling between the domains is minimized. In a typical anisotropic Navier-Stokes grid, this corresponds to domain partition boundaries which span the weak (streamwise) connections of the grid. This approach has been pursued to some extent using an ILU preconditioned solver on the local domains in the context of a multigrid algorithm in reference [46]. For parallel computations, in the context of a Schwarz method for example, this constrained partitioning approach would need to balance the potential for increased convergence versus the possible increased communication volume generated by such partitions.

Directional coarsening methods, where coarse multigrid levels are constructed by removing points based on the relative strength of local stencil coefficients, is an integral part of any algebraic multigrid method [49]. Directional coarsening methods have also been applied in the context of geometric multigrid methods [39], and agglomeration multigrid methods [33, 46, 13]. These approaches enable the construction of coarse grid levels which are optimal for the given problem. In most cases, convergence rates which are independent of the degree of anisotropy can be achieved.

One of the drawbacks of these methods is that, while good convergence rates can be achieved, the memory and cpu time associated with each multigrid cycle may increase substantially when directional coarsening is employed. This is due to the higher complexity of the directionally coarsened levels as compared with isotropically coarsened levels. In the latter case, grid complexity decreases by a factor of 4 in 2D and 8 in 3D when going to each next coarser level, while in the case of pure uni-directional coarsening (i.e. semi-coarsening), the complexity reduction is only a factor of 2 both in 2D and 3D. For this reason, aggressive directional coarsening strategies have been advocated, where the coarsening proceeds in the required direction, but at an accelerated rate, by removing more than just nearest neighbor grid points between two consecutive levels. Figure 4.9 depicts an aggressive directional agglomeration in the leading-edge vicinity of the mesh of Figure 4.8. The thick lines represent the groupings of fine grid vertices into coarse level sets or control-volumes. In highly stretched regions of the mesh, the directional grouping of up to four consecutive vertices is observed. In order to preserve favorable

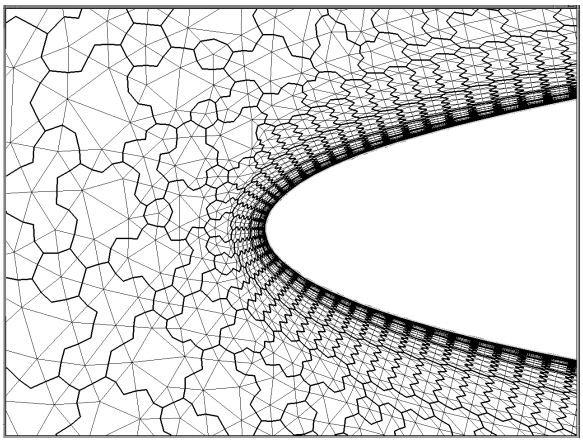

FIG. 4.9. *Agglomerated multigrid level constructed on grid of Figure 4.8 illustrating aggressive 4:1 directional coarsening in boundary layer regions (reproduced from [35])*

convergence rates, the underlying explicit smoothers used in the multigrid scheme must be upgraded to include a locally implicit behavior [46, 34].

In references [34, 35], a line-implicit preconditioned multi-stage scheme has been used in conjunction with a directional coarsening agglomeration multigrid strategy. Figure 4.10 compares the convergence rates achieved by this algorithm with the convergence of the baseline isotropic explicit multigrid algorithm, for three different grids. The three grids contain identical streamwise resolution but a varying degree of stretching in the boundary-layer and wake regions. The first grid contains a normal wall spacing of 10^{-5} chords, while the second and third grids contain normal wall spacings of 10^{-6} and 10^{-7} chords respectively. Each subsequent grid therefore exhibits an order of magnitude higher stretching than the previous grid. The middle grid corresponds to the grid depicted in Figure 4.8 and the implicit lines used by the directional implicit multigrid algorithm are those depicted in the same figure, while the first level directional agglomeration corresponds to the graph depicted in Figure 4.9. While the isotropic explicit multigrid algorithm convergence rates decay with increasing grid stretching, the directional-implicit multigrid algorithm is seen to be relatively insensitive to the amount of grid stretching, as depicted in Figure 4.10, providing rapid convergence to machine zero in under 600 cycles for all cases.

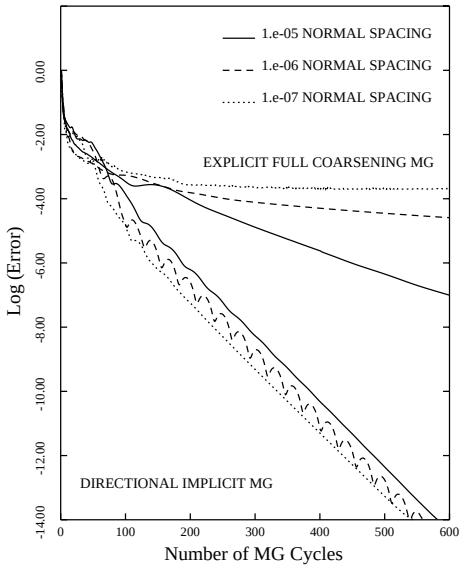

FIG. 4.10. *Comparison of convergence rates for isotropic-explicit and directional-implicit multigrid schemes on three grids of varying normal resolution*

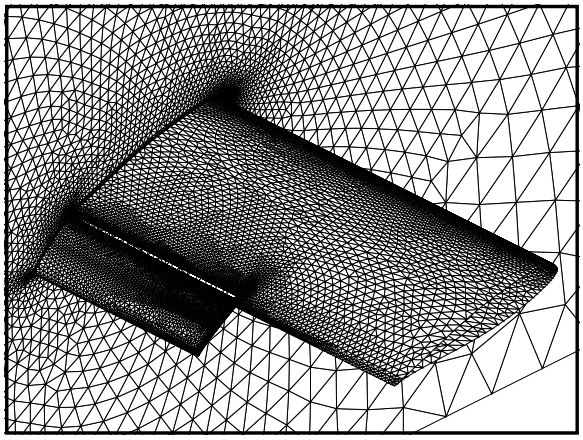

FIG. 4.11. *Stretched unstructured grid on partial-span flap geometry. Number of vertices = 549,176; wall spacing* = 10^{-5} *chords*

This algorithm has also been applied to three-dimensional problems using one-dimensional implicit lines normal to the body surfaces in boundary-layer regions constructed with a weighted graph algorithm. The turbulent flow over a partial span flap wing geometry on a grid of 549,176 vertices depicted in Figure 4.11 has been computed for a freestream Mach number of 0.2, an incidence of 10 degrees, and a Reynolds number of 3.2 million [35]. The convergence rates of the three-dimensional isotropic explicit multigrid algorithm and the directional-implicit multigrid algorithm are compared in Figure 4.12. In this case, the directional-implicit multigrid algorithm provides substantial speedup over the isotropic explicit multigrid algorithm, but falls short of the convergence rates observed in two-dimensional problems (c.f. Figure 4.10). When a GMRES algorithm is added to the solver, using the previous solver as a GMRES(20) preconditioner, (using 20 search directions) substantial improvement in the asymptotic convergence rate is observed, achieving a total of 9

orders of magnitude convergence in 600 multigrid cycles. While the overall computation time per cycle varies only slightly between the various schemes depicted in Figure 4.12, the increase in convergence due to the GMRES algorithm comes at a 50 % increase in memory usage due to the storage of the search directions for the GMRES procedure.

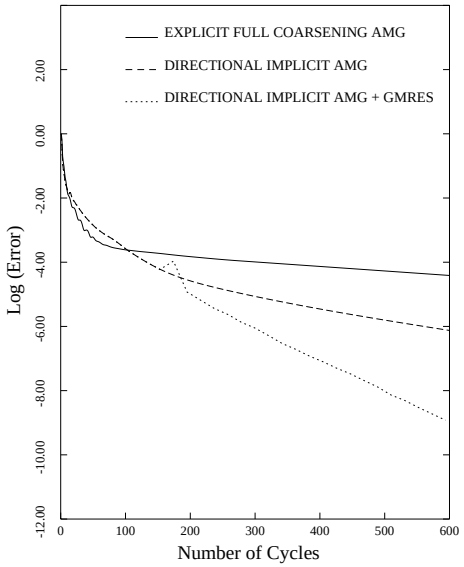

FIG. 4.12. *Convergence rates attained by three different multigrid schemes for flow over partial-span flap geometry*

5. The Potential for Future Advances. While much progress has been achieved over the last decade in developing efficient solution strategies for unstructured mesh problems, the best convergence rates currently available on CFD problems remain one to two orders of magnitude slower than what can be achieved by the best algorithms on simple linear elliptic problems.

For example, a properly formulated multigrid method is capable of producing convergence rates of 0.1 for a Poisson equation, meaning that for each multigrid cycle, the error is reduced by an order of magnitude. This enables the solution of such problems in less than 10 multigrid cycles. By contrast, the best multigrid methods for CFD problems still require several hundred cycles to achieve acceptable convergence levels.

Optimal multigrid algorithms for CFD problems, capable of convergence rates of the order of 0.1, have recently been demonstrated for simple two-dimensional incompressible and compressible flow problems both on structured and unstructured meshes [8, 54, 48]. The central idea in the development of these schemes is the decoupling of the hyperbolic and elliptic parts of the governing equations. The decoupled hyperbolic convective terms are solved by sweeping through the domain in the downstream direction, while the elliptic part, which corresponds to a pressure Poisson equation, is solved by an optimally convergent multigrid method. As an example, the flow over a bump in a channel, displayed in Figure 5.1, has been computed on several unstructured grids of increasing resolution (obtained by subdividing a corresponding structured grid) in reference [48]. Figure 5.2 depicts the convergence rates obtained by this approach, which are essentially grid independent, and realize a 6 to 7 order of reduction in the residuals in only 10 multigrid cycles.

One of the problems with these approaches, is that they are not simply convergence acceleration techniques which can be applied to existing discretizations, but require the use of a particular discretization

which in turn enables the decoupling into elliptic and hyperbolic parts. In this sense, they are related to the local-preconditioning methods described previously, which in addition to modifying the time-stepping behavior of the corresponding un-preconditioned scheme, result in changes to the discretization. However, in the present cases, the changes to the discretization represent completely different approaches from the traditional conservative-variable finite-volume or finite-element discretizations typically employed in existing CFD approaches.

This may severely limit their applicability, since it may not be feasible in general to radically overhaul familiar validated discretizations in existing codes. A potential alternative is the development of a preconditioning matrix which, when applied to existing discretizations, mimicks the separation of elliptic and hyperbolic terms in the time-stepping scheme.

While the treatment of the convective terms using a sweeping method has proved to be very successful for simple flows of the type displayed in Figure 5.1, the extension of this approach to more complex flows involving regions of rotating, recirculating and reverse flow remains an area of research.

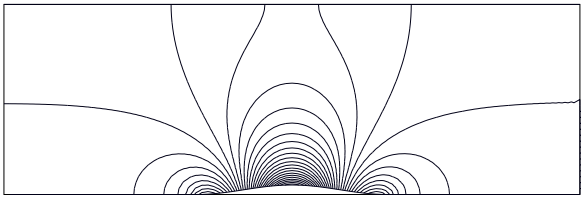

FIG. 5.1. *Pressure contours computed for flow over bump on unstructured grid of* 97×33 *vertices (reproduced from reference [48])*

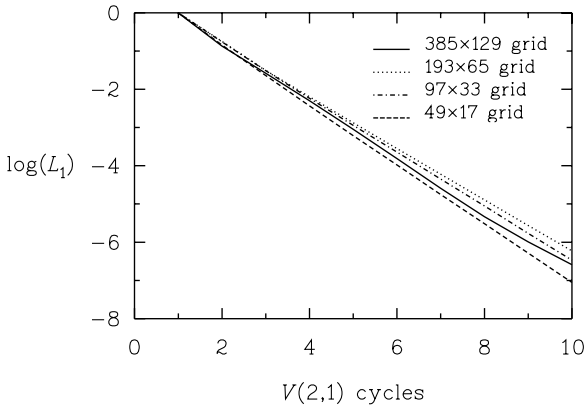

FIG. 5.2. *Demonstration of textbook multigrid convergence for computation of inviscid flow over bumb on unstructured grids of widely varying resolution (reproduced from reference [48])*

6. Conclusions. Advances in our ability to solve large complex stiff problems are likely to come as much from increases in algorithmic efficiency as from increases in hardware capability over the next decade. Of particular interest is the development of algorithms which are well suited to the current trends in computer hardware, which include parallelism and memory-latency tolerance. There exist a wide variety of techniques for enhancing convergence, and most successful solution strategies employ combinations of several of these techniques. The optimal algorithm is a function not only of the target hardware platform, but also of the characteristics of the desired application, the usability of the final method, and its robustness. Thus, further

development on many different types of algorithms is likely to persist in the future. The development of solution methods capable of converging problems in the same number of steps as a direct (Newton) method, but at little additional cost per iteration over a purely explicit scheme remains a long-term goal.

7. Acknowledgements. The contributed results of David Keyes, ICASE and Old Dominion University, Kyle Anderson, NASA Langley Research Center, and Tom Roberts, NASA Langley Research Center are greatly acknowledged.

REFERENCES

[1] *Grand Chanllenges: High-performance computing and communications.* A report by the committee on physical, mathematical, and engineering sciences, Federal Coordinating Council for Science, Engineering and Technology, Office of Science and Technology Policy, 1992.

[2] http://www.cray.com/products/systems/crayt3e/1200/ performance.html, 1998.

[3] W. K. Anderson, *A grid generation and flow solution method for the Euler equations on unstructured grids*, Journal of Computational Physics, 110 (1994), pp. 23–38.

[4] T. J. Barth, *Numerical aspects of computing viscous high Reynolds number flows on unstructured meshes.* AIAA Paper 91-0721, Jan. 1991.

[5] ——, *Parallel CFD algorithms on unstructured meshes*, in Special Course on Parallel Computing in CFD, May 1995, pp. 7–1,7–41. AGARD Report-807.

[6] T. J. Barth and S. W. Linton, *An unstructured mesh Newton solver for compressible fluid flow and its parallel implementation.* AIAA paper 95-0221, Jan. 1995.

[7] J. T. Batina, *Implicit upwind solution algorithms for three-dimensional unstructured meshes*, AIAA J., 31 (1993), pp. 801–805.

[8] A. Brandt and I. Yavneh, *Accelerated multigrid convergence and high-Reynolds number recirculating flows*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 607–627.

[9] X. Cai, D. E. Keyes, and V. Venkatakrishnan, *Newton-Krylov-Schwarz: An implicit solver for CFD*, in Proceedings of Eighth International Conference on Domain Decomposition Methods (R. Glowinski et al., eds.), Wiley, New York, 1997, pp. 387–400.

[10] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, Acta Numerica, (1994), pp. 61–143.

[11] Y. Choi and C. Merkle, *The application of preconditioning to viscous flows*, J. Comp. Phys., 105 (1993), pp. 207–223.

[12] E. Cuthill and J. McKee, *Reducing the band width of sparse symmetric matrices*, in Proc. ACM Nat. Conference, 1969, pp. 157–172.

[13] J. Francescatto, *Résolution de l'équation de Poisson sur des maillages étirés par une methode multigrille.* INRIA Report No. 2712, Nov. 1995.

[14] J. A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1981.

[15] H. Guillard, *Node nested multigrid with Delaunay coarsening.* INRIA Report No. 1898, 1993.

[16] A. Haselbacher, J. McGuirk, and G. Page, *Finite-volume discretization aspects for viscous flows on unstructured meshes*, in Proceedings of the 13th AIAA CFD Conference, Snowmass, CO, June 1997, pp. 599–609. AIAA Paper 97-1946-CP.

[17] O. Hassan, K. Morgan, and J. Peraire, *An adaptive implicit/explicit finite element scheme for compressible high speed flows*. AIAA Paper 89-0363, Jan. 1989.

[18] B. Hendrickson and R. Leland, *A multilevel algorithm for partitioning graphs*, in Proceedings Proc. Supercomputing '95, ACM, Dec. 1995.

[19] B. R. Hutchinson, P. F. Galpin, and G. D. Raithby, *Application of additive correction multigrid to the coupled fluid flow equations*, Numerical Heat Transfer, 13 (1988), pp. 133–147.

[20] B. R. Hutchinson and G. D. Raithby, *A multigrid method based on the additive correction strategy*, Numerical Heat Transfer, 9 (1986), pp. 511–537.

[21] A. Jameson, *Solution of the Euler equations by a multigrid method*, Applied Mathematics and Computation, 13 (1983), pp. 327–356.

[22] A. Jameson and S. Yoon, *Lower-upper implicit schemes with multiple grids for the Euler equations*, AIAA Journal, 25 (1987), pp. 929–935.

[23] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, Tech. Report Technical Report 95-035, University of Minnesota, 1995. A short version appears in Intl. Conf. on Parallel Processing 1995.

[24] D. E. Keyes, D. K. Kaushik, and B. F. Smith, *Prospects for CFD on petaflops systems*, in CFD Review M. Hafez, et. al., eds., Wiley, New York, 1997.

[25] B. Koobus, M. H. Lallemand, and A. Dervieux, *Unstructured volume-agglomeration MG: Solution of the poisson equation*. INRIA Report 1946, June 1993.

[26] B. Koren, *Defect correction and multigrid for an efficient and accurate computation of airfoil flows*, Journal of Computational Physics, 77 (1988), pp. 183–206.

[27] M. Lallemand, H. Steve, and A. Dervieux, *Unstructured multigridding by volume agglomeration: Current status*, Computers and Fluids, 21 (1992), pp. 397–433.

[28] M. P. Leclercq, *Resolution des Equations d'Euler par des Methods Multigrilles Conditions aux Limites en Regime Hypersonique*, PhD thesis, Univerite de Saint-Etienne, Applied Math, Apr. 1990.

[29] B. v. W. T. Lee and P. Roe, *Characteristic time-stepping or local preconditioning of the Euler equations*, in Proceedings of the 10th AIAA CFD Conference, Honolulu, Hawaii, June 1991, pp. 260–282. AIAA Paper 91-1552-CP.

[30] R. Lohner, *Renumbering strategies for unstructured grid solvers operating on shared-memory cache-based parallel machines*, in Proceedings of the 13th AIAA CFD Conference, Snowmass, CO, June 1997, pp. 1015–1025. AIAA Paper 97-2045-CP.

[31] H. Luo, J. D. Baum, R. Löhner, and J. Cabello, *Adaptive edge-based finite element schemes for the Euler and Navier-Stokes equations on unstructured grids*. AIAA Paper 93-0336, Jan. 1993.

[32] D. Martin and Löhner, *An implicit linelet-based solver for incompressible flows*. AIAA Paper 92-0668, Jan. 1992.

[33] D. J. Mavriplis, *Multigrid techniques for unstructured meshes*, in VKI Lecture Series VKI-LS 1995-02, Mar. 1995.

[34] ——, *Multigrid strategies for viscous flow solvers on anisotrpic unstructured meshes*, in Proceedings of the 13th AIAA CFD Conference, Snowmass, CO, June 1997, pp. 659–675. AIAA Paper 97-1952-CP.

[35] ——, *Directional agglomeration multigrid techniques for high-Reynolds number viscous flows*. AIAA paper 98-0612, Jan. 1998.

[36] ——, *Threee dimensional high-lift analysis using a parallel unstructured multigrid solver*. AIAA paper 98-2619, to be presented at the 16th AIAA Applied Aerodynamics Conference, Albuquerque, NM,

June 1998.

[37] D. J. Mavriplis and V. Venkatakrishnan, *Agglomeration multigrid for two dimensional viscous flows*, Computers and Fluids, 24 (1995), pp. 553–570.

[38] ——, *A unified multigrid solver for the Navier-Stokes equations on mixed element meshes*, International Journal for Computational Fluid Dynamics, 8 (1997), pp. 247–263.

[39] E. M. D. J. Mavriplis and V. Venkatakrishnan, *Coarsening strategies for unstructured multigrid techniques with application to anisotropic problems*, in 7th Copper Mountain Conf. on Multigrid Methods, Apr. 1995, pp. 591–606. NASA Conference Publication 3339.

[40] E. Morano and A. Dervieux, *Looking for O(N) Navier-Stokes solutions on non-structured meshes*, in 6th Copper Mountain Conf. on Multigrid Methods, 1993, pp. 449–464. NASA Conference Publication 3224.

[41] C. Ollivier-Gooch, *Towards problem-independent multigrid convergence rates for unstructured mesh methods i: Inviscid and laminar flows*, in Proceedings of the 6th International Symposium on CFD, Lake Tahoe, NV, Sept. 1995.

[42] J. Peraire, J. Peiró, and K. Morgan, *A 3D finite-element multigrid solver for the Euler equations.* AIAA Paper 92-0449, Jan. 1992.

[43] N. Pierce and M. Giles, *Preconditioning on stretched meshes.* AIAA paper 96-0889, Jan. 1996.

[44] A. Pothen, H. D. Simon, and K. P. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 430–452.

[45] W. K. A. R. Rausch and D. Bonhaus, *Implicit multigrid algorithms for incompressible turbulent flows on unstructured grids*, in Proceedings of the 12th AIAA CFD Conference, San Diego CA, June 1995. AIAA Paper 95-1740-CP.

[46] M. Raw, *Robustness of coupled algebraic multigrid for the Navier-Stokes equations.* AIAA paper 96-0297, Jan. 1996.

[47] K. Riemslagh and E. Dick, *A multigrid method for steady Euler equations on unstructured adaptive grids*, in 6th Copper Mountain Conf. on Multigrid Methods, NASA conference publication 3224, 1993, pp. 527–542.

[48] T. W. Roberts, D. Sidilkover, and R. C. Swanson, *Textbook multigrid efficiency for the steady-state Euler equations*, in Proceedings of the 13th AIAA CFD Conference, Snowmass, CO, June 1997, pp. 629–647. AIAA Paper 97-1949-CP.

[49] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., SIAM Frontiers in Applied Mathematics, Philadelphia, 1987, SIAM, pp. 73–131.

[50] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Series in Computer Science, PWS Publishing Company, Boston, MA, 1996.

[51] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[52] D. Sharov and K. Nakahashi, *Reordering of 3D hybrid unstructured grids for vectrorized LU-SGS Navier-Stokes computations*, in Proceedings of the 13th AIAA CFD Conference, Snowmass, CO, June 1997, pp. 131–138. AIAA Paper 97-2102-CP.

[53] W. A. Smith, *Multigrid solution of transonic flow on unstructured grids*, in Recent Advances and Applications in Computational Fluid Dynamics, Nov. 1990. Proceedings of the ASME Winter Annual Meeting, Ed. O. Baysal.

[54] S. Ta'assan, *Canonical-variables method multigrid for steady-state euler equations.* ICASE Report

94-14, 1994.

[55] J. L. Thomas, *A perspective on computational fluid dynamics research at NASA*, in Numerical Methods for Fluid Dynamics V, Oxford, 1995, pp. 19–36.

[56] J. L. Thomas, D. L. Bonhaus, and W. K. Anderson, *An (O)(nm^2) plane solver for thge compressible Navier-Stokes equations*. Abstract submitted to the 37th AIAA Aerospace Sciences Meeting, Reno NV, Jan. 1999.

[57] E. Turkel, *Preconditioning methods for solving the incompressible and low speed compressible equations*, J. Comp. Phys., 72 (1987), pp. 277–298.

[58] V. Venkatakrishnan, *Improved multigrid performance of navier-stokes solvers*. AIAA Paper 98-2967, June 1998.

[59] V. Venkatakrishnan and T. J. Barth, *Application of direct solvers to unstructured meshes for the Euler and Navier-Stokes equations using upwind schemes*. AIAA Paper 89-0364, Jan. 1989.

[60] V. Venkatakrishnan and D. J. Mavriplis, *Implicit solvers for unstructured meshes*, Journal of Computational Physics, 105 (1993), pp. 83–91.

[61] E. J. N. W. K. A. R. W. Walters and D. E. Keyes, *Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code*, in Proceedings of the 12th AIAA CFD Conference, San Diego CA, June 1995. AIAA Paper 95-1733-CP.

[62] J. Weiss, J. Maruszewski, and W. Smith, *Implicit solution of the Navier-Stokes equations on unstructured meshes*, in Proceedings of the 13th AIAA CFD Conference, Snowmass, CO, June 1997, pp. 139–149. AIAA Paper 97-2103-CP.

[63] J. M. Weiss and W. A. Smith, *Preconditioning applied to variable and constant density time-accurate flows on unstructured meshes*. AIAA Paper 94-2209, June 1994.

[64] D. L. Whitaker, *Three-dimensional unstructured grid Euler computations using a fully-implicit, upwind method*. AIAA Paper 93-3337CP, July 1993.

[65] L. B. W. N. J. Yu and D. P. Young, *GMRES acceleration of computational fluid dynamic codes*, in Proceedings of the 7th AIAA CFD Conference, July 1985, pp. 67–74. AIAA Paper 85-1494-CP.

[66] D. Zingg and A. Pueyo, *An efficient Newton-GMRES solver for aerodynamic computations*, in Proceedings of the 13th AIAA CFD Conference, Snowmass, CO, June 1997, pp. 712–721. AIAA Paper 97-1955-CP.